EXPRESS MAIL NO. EV301024810US

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicants: Yolanta BERESNEVICHIENE, et al.
Serial No.: Not yet assigned
Filed: Concurrently herewith
For: "IMPROVEMENTS IN AND RELATING TO DATA HANDLING APPARATUS AND METHODS"

Our Ref: B-5328 621562-9

Date: January 26, 2004

CLAIM TO PRIORITY UNDER 35 U.S.C. 119

MAIL STOP PATENT APPLICATION
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Sir:

[X] Applicants hereby make a right of priority claim under 35 U.S.C. 119 for the benefit of the filing date(s) of the following corresponding foreign application(s):

COUNTRY	FILING DATE	SERIAL NUMBER
GB	27 January 2003	0301779.5

[ ] A certified copy of each of the above-noted patent applications was filed with the Parent Application No._____.

[X] To support applicants' claim, a certified copy of the above-identified foreign patent application is enclosed herewith.

[ ] The priority document will be forwarded to the Patent Office when required or prior to issuance.

Respectfully submitted,

Richard P. Berg
Attorney for Applicant
Reg. No. 28,145

LADAS & PARRY
5670 Wilshire Boulevard
Suite 2100
Los Angeles, CA 90036
Telephone: (323) 934-2300
Telefax: (323) 934-0202

EV301024810 US

INVESTOR IN PEOPLE

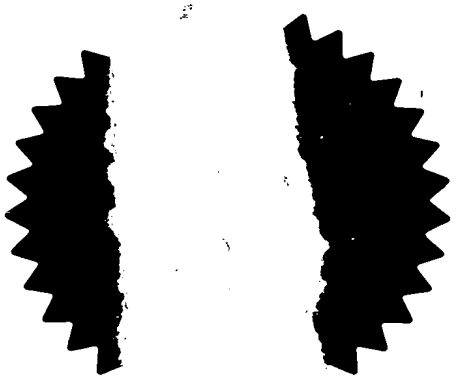
The Patent Office
Concept House
Cardiff Road
Newport
South Wales
NP10 8QQ

I, the undersigned, being an officer duly authorised in accordance with Section 74(1) and (4) of the Deregulation & Contracting Out Act 1994, to sign and issue certificates on behalf of the Comptroller-General, hereby certify that annexed hereto is a true copy of the documents as originally filed in connection with the patent application identified therein.

In accordance with the Patents (Companies Re-registration) Rules 1982, if a company named in this certificate and any accompanying documents has re-registered under the Companies Act 1980 with the same name as that with which it was registered immediately before re-registration save for the substitution as, or inclusion as, the last part of the name of the words "public limited company" or their equivalents in Welsh, references to the name of the company in this certificate and any accompanying documents shall be treated as references to the name with which it is so re-registered.

In accordance with the rules, the words "public limited company" may be replaced by p.l.c., plc, P.L.C. or PLC.

Re-registration under the Companies Act does not constitute a new legal entity but merely subjects the company to certain additional company law rules.



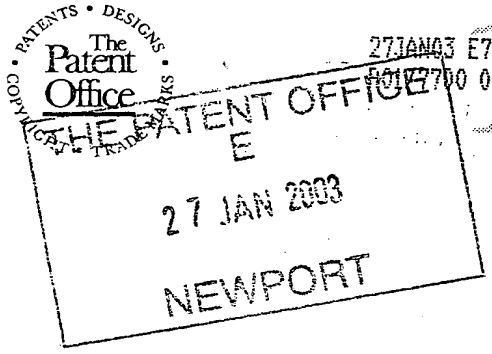
Signed 

Dated 12 December 2003

EV 301024810US

Patents Form 1/77

Patents Act 1977
(Rule



27JAN03 E780039-1 D01463
P01/7700 0.00-0301779.5

1/77

Request for grant of a patent

(See the notes on the back of this form. You can also get an explanatory leaflet from the Patent Office to help you fill in this form)

The Patent Office

Cardiff Road
Newport
South Wales
NP10 8QQ

1. Your reference — 200207541-1 GB

2. Patent application number
(The Patent Office will fill in this part) — 0301779.5

3. Full name, address and postcode of the or of each applicant *(underline all surnames)*

Hewlett-Packard Company
3000 Hanover Street
Palo Alto
CA 94304, USA

Patents ADP number *(if you know it)*

If the applicant is a corporate body, give the country/state of its incorporation — Delaware, USA

7812985001

4. Title of the invention — Improvements in and Relating to Data Handling Apparatus and Methods

5. Name of your agent *(if you have one)*

"Address for service" in the United Kingdom to which all correspondence should be sent *(including the postcode)*

Richard A. Lawrence
Hewlett-Packard Ltd, IP Section
Filton Road, Stoke Gifford
Bristol BS34 8QZ

Patents ADP number *(if you know it)* — 7448038001

6. If you are declaring priority from one or more earlier patent applications, give the country and the date of filing of the or of each of these earlier applications and *(if you know it)* the or each application number

Country	Priority application number *(if you know it)*	Date of filing *(day / month / year)*

7. If this application is divided or otherwise derived from an earlier UK application, give the number and the filing date of the earlier application

Number of earlier application	Date of filing *(day / month / year)*

8. Is a statement of inventorship and of right to grant of a patent required in support of this request? *(Answer 'Yes' if:*

a) any applicant named in part 3 is not an inventor, or
b) there is an inventor who is not named as an applicant, or
c) any named applicant is a corporate body.
See note (d))

Yes

9. Enter the number of sheets for any of the following items you are filing with this form. Do not count copies of the same document

Continuation sheets of this form	
Description	26
Claim(s)	7
Abstract	1
Drawing(s)	7 + 7

10. If you are also filing any of the following, state how many against each item.

Priority documents	-
Translations of priority documents	-
Statement of inventorship and right to grant of a patent *(Patents Form 7/77)*	1
Request for preliminary examination and search *(Patents Form 9/77)*	1
Request for substantive examination *(Patents Form 10/77)*	-
Any other documents *(please specify)*	Fee Sheet

11. I/We request the grant of a patent on the basis of this application.

Signature Richard A. Lawrence Date 27/1/03

12. Name and daytime telephone number of person to contact in the United Kingdom Meg Joyce Tel: 0117-312-9068

Warning

After an application for a patent has been filed, the Comptroller of the Patent Office will consider whether publication or communication of the invention should be prohibited or restricted under Section 22 of the Patents Act 1977. You will be informed if it is necessary to prohibit or restrict your invention in this way. Furthermore, if you live in the United Kingdom, Section 23 of the Patents Act 1977 stops you from applying for a patent abroad without first getting written permission from the Patent Office unless an application has been filed at least 6 weeks beforehand in the United Kingdom for a patent for the same invention and either no direction prohibiting publication or communication has been given, or any such direction has been revoked.

Notes

a) *If you need help to fill in this form or you have any questions, please contact the Patent Office on 08459 500505.*

b) *Write your answers in capital letters using black ink or you may type them.*

c) *If there is not enough space for all the relevant details on any part of this form, please continue on a separate sheet of paper and write "see continuation sheet" in the relevant part(s). Any continuation sheet should be attached to this form.*

d) *If you have answered 'Yes' Patents Form 7/77 will need to be filed.*

e) *Once you have filled in the form you must remember to sign and date it.*

f) *For details of the fee and ways to pay please contact the Patent Office.*

Improvements in and Relating to Data Handling Apparatus and Methods

The present invention relates to data handling apparatus and methods, to computer programs for implementing such methods and to computing platforms configured to operate according to such methods.

Data management is increasingly important as widespread access to public computer networks facilitates distribution of data. Distribution of data over public computer networks may be undesirable when the data in question comprises sensitive, confidential, copyright or other similar information.

A computer operating system can typically monitor input of data to a process or output of data by a process and apply appropriate management restrictions to these operations. Exemplary restrictions may prevent write operations to a public network, or to external memory devices for data having certain identifiable characteristics. However, manipulation of data within a process can not be monitored by the operating system. Such manipulation may modify the identifiable characteristics of data, and thus prevent the operating system from carrying out effective data management.

Particular problems arise when different types of data are assigned different levels of restriction, and processes involving data from different levels of restriction are run alongside one another. An operating system cannot guarantee that the different types of data have not been mixed. To maintain a desired level of restriction for the

most restricted data in these circumstances, this level of restriction must be applied to all data involved in the processes. Consequently, data can only be upgraded to more restricted levels, leading to a system in which only highly trusted users/systems are allowed access to any data.

In prior art systems, security policies are applied at the application level, thus meaning that each application requires a new security policy module dedicated to it.

It is an aim of preferred embodiments of the present invention to overcome at least some of the problems associated with the prior art, whether identified herein, or otherwise.

According to the present invention in a first aspect, there is provided a data handling apparatus for a computer platform using an operating system, the apparatus comprising a system call monitor for detecting predetermined system calls, and means for applying a data handling policy to the system call upon a predetermined system call being detected.

Using such an apparatus, because the security policy determination is initiated at the operating system level by monitoring system calls, it can be made application independent. So, for instance, on a given platform it would not matter which e-mail application is being used, the data handling apparatus could control data usage.

Suitably, in which the policy is to require the encryption of at least some of the data.

Suitably, a policy interpreter in its application of the policy automatically encrypts the at least some of the data.

Suitably, predetermined system calls are those involving the transmission of data externally of the computing platform.

Suitably, the means for applying a data handling policy comprises a tag determiner for determining any security tags associated with data handled by the system call, and a policy interpreter for determining a policy according to any such tags and for applying the policy.

Suitably, the policy interpreter is configured to use the intended destination of the data as a factor in determining the policy for the data.

Suitably, the policy interpreter comprises a policy database including tag policies and a policy reconciler for generating a composite policy from the tag policies relevant to the data.

Suitably, the computing platform comprises a data management unit, the data management unit arranged to associate data management information with data input to a process, and regulate operating system operations involving the data according to the data management information.

Suitably, the computing platform further comprises a memory space, and is arranged to load the process into the

memory space and run the process under the control of the data management unit.

Suitably, the data management information is associated with at least one data sub-unit as data is input to a process from a data unit comprising a plurality of sub-units.

Suitably, data management information is associated with each independently addressable data unit.

Suitably, the data management unit comprises part of an operating system kernel space.

Suitably, the operating system kernel space comprises a tagging driver arranged to control loading of a supervisor code into the memory space with the process.

Suitably, the supervisor code controls the process at run time to administer the operating system data management unit.

Suitably, the supervisor code is arranged to analyse instructions of the process to identify operations involving the data, and, provide instructions relating to the data management information with the operations involving the data.

Suitably, the memory space further comprises a data management information area under control of the supervisor code arranged to store the data management information.

Suitably, the data management unit comprises a data filter to identify data management information associated with data that is to be read into the memory space.

Suitably, the data management unit further comprises a tag management module arranged to allow a user to specify data management information to be associated with data.

Suitably, the data management unit comprises a tag propagation module arranged to maintain an association with the data that has been read into the process and the data management information associated therewith.

Suitably, the tag propagation module is arranged to maintain an association between an output of operations carried out within the process and the data management information associated with the data involved in the operations.

Suitably, the tag propagation module comprises state machine automatons arranged to maintain an association between an output of operations carried out within the process and the data management information associated with the data involved in the operations.

According to the present invention in a second aspect, there is provided a data handling method for a computer platform using an operating system, the method comprising the steps of: detecting predetermined system calls, and applying a data handling policy to the system call upon a predetermined system call being detected.

Suitably, the policy is to require the encryption of at least some of the data.

Suitably, in its application of the policy at least some of the data is automatically encrypted.

Suitably, predetermined system calls are those involving the transmission of data externally of the computing platform.

Suitably, the method includes the steps of: determining any security tags associated with data handled by the system call, determining a policy according to any such tags and applying the policy.

Suitably, a composite policy is generated from the tag policies relevant to the data.

Suitably, the intended destination of the data is used as a factor in determining the policy for the data.

Suitably, the method further comprises the steps of: (a) associating data management information with data input to a process; and (b) regulating operating system operations involving the data according to the data management information.

Suitably, supervisor code administers the method by controlling the process at run time.

Suitably, the step (a) comprises associating data management information with data as the data is read into a memory space.

Suitably, the step (a) comprises associating data management information with at least one data sub-unit as data is read into a memory space from a data unit comprising a plurality of data sub-units.

Suitably, the step (a) comprises associating data management information with each independently addressable data unit that is read into the memory space.

Suitably, the data management information is written to a data management memory space under control of the supervisor code.

Suitably, the supervisor code comprises state machine automatons arranged to control the writing of data management information to the data management memory space.

Suitably, the step (b) comprises sub-steps (b1) identifying an operation involving the data; (b2) if the operation involves the data and is carried out within the process, maintaining an association between an output of the operation and the data management information; and (b3) if the operation involving the data includes a write operation to a location external to the process, selectively performing the operation dependent on the data management information.

Suitably, the step (b1) comprises: analysing process instructions to identify operations involving the data; and, providing instructions relating to the data

management information with the operations involving the data.

Suitably, the process instructions are analysed as blocks, each block defined by operations up to a terminating condition.

According to the present invention in a third aspect, there is provided a computer program for controlling a computing platform to operate in accordance with the second aspect of the invention.

According to the present invention in a fourth aspect, there is provided a computer platform configured to operate according with the second aspect of the invention.

For a better understanding of the invention, and to show how embodiments of the same may be carried into effect, reference will now be made, by way of example, to the accompanying diagrammatic drawings in which:

Figure 1 shows a computing platform for computer operating system data management according to the present invention;

Figure 2 shows a first operating system data management architecture suitable for use in the computing platform of Figure 1;

Figure 3 shows a second operating system data management architecture suitable for use in the computing platform of Figure 1; and

Figure 4 shows a flow diagram comprising steps involved in operation of the above described figures;

Figure 5 shows a flow diagram comprising further steps involved as part of the Figure 4 operation;

Figure 6 shows a data handling apparatus according to the present invention;

Figure 7 shows a functional flow diagram of a method of operation of the apparatus of Figure 6; and

Figure 8 shows a functional flow diagram of part of the method of Figure 7.

Data management in the form of data flow control can offer a high degree of security for identifiable data. Permitted operations for identifiable data form a security policy for that data. However, security of data management systems based on data flow control is compromised if applications involved in data processing can not be trusted to enforce the security policies for all data units and sub-units to which the applications have access. In this document, the term "process" relates to a computing process. Typically, a computing process comprises the sequence of states run through by software as that software is executed.

Figure 1 shows a computing platform 1 for computer operating system data management comprising, a processor 5, a memory space 10, an OS kernel space 20 comprising a data management unit 21 and a disk 30. The memory space 10 comprises an area of memory that can be addressed by

user applications. The processor 5 is coupled to the memory space 10 and the OS kernel space 20 by a bus 6. In use, the computing platform 1 loads a process to be run on the processor 5 from the disk 30 into the memory space 10. It will be appreciated that the process to be run on the processor 5 could be loaded from other locations. The process is run on the processor under the control of the data management unit 21 such that operations involving data read into the memory space 10 by the process are regulated by the data management unit 21. The data management unit 21 regulates operations involving the data according to data management information associated with the data as it is read into the memory space 10.

The data management unit 21 propagates the data management information around the memory space 10 as process operations involving that data are carried out, and prevents the data management information from being read or written over by other operations. The data management unit includes a set of allowable operations for data having particular types of data management information therewith. By inspecting the data management information associated with a particular piece of data, the data management unit 21 can establish whether a desired operation is allowed for that data, and regulate the process operations accordingly.

Figure 2 shows an example operating system data management architecture comprising an OS kernel space and a memory space suitable for use in the computing platform of Figure 1. The example architecture of Figure 2 enables regulation of operations involving data read into a memory space by enforcing data flow control on applications using

that data. The example architecture of Figure 2 relates to the Windows NT operating system. Windows NT is a registered trade mark of Microsoft Corporation.

Figure 2 shows a memory space comprising a user space 100 and an OS kernel space 200. The user space 100 comprises application memory spaces 110A,110B, supervisor code 120A,120B, and a tag table 130. The OS kernel space 200 comprises a standard NT kernel 250, file system driver 202 and storage device drivers 203. The OS kernel space 200 further comprises a tagging driver 210, a tag propagation module 220, and a tag management module 230 and a data filter 240.

When an application is to be run in the user space 100, information comprising the application code along with any required function libraries, application data etc. is loaded into a block of user memory space comprising the application memory space 110 under the control of the NT kernel 250. The tagging driver 210 further appends supervisor code to the application memory space 110 and sets aside a memory area for data management information. This memory area comprises the tag table 130.

In preference to allowing the NT kernel 250 to run the application code, the tagging driver 210 receives a code execution notification from the NT kernel 210 and runs the supervisor code 120

When run, the supervisor code 120 scans the application code starting from a first instruction of the application code, and continues through the instructions of the application code until a terminating condition is reached.

A terminating condition comprisesan instruction that causes a change in execution flow of the application instructions., Example terminating conditions include jumps to a subroutines, interrupts etc. A portion of the application code between terminating conditions comprises a block of code.

The block of code is disassembled, and data management instructions are provided for any instructions comprising data read/writes to the memory, disk, registers or other functional units such as logic units, or to other input/output (I/O) devices. The data management instructions may include the original instruction that prompted provision of the data management instructions, along with additional instructions relating to data management. Once a block of the application code has been scanned and modified, the modified code can be executed. The scanning process is then repeated, starting with the first instruction of the next block.

At a first system call of the application code relating to a particular piece of data, typically a read instruction, the first data management instruction associates data management information with the data. The data management information comprises a tag held in the tag table 130. The tag table 130 comprises a data management information memory area which can only be accessed by the supervisor code 120. Preferably, a tag is applied to each independently addressable unit of data - normally each byte of data. By applying a tag to each independently addressable piece of data all useable data is tagged, and, maximum flexibility regarding the association of data with

a tag is maintained. A tag may preferably comprise a byte or other data unit.

A tag identifies a data management policy to be applied to the data associated with that tag. Different data management policies may specify a number of rules to be enforced in relation to data under that data management policy, for example, "data under this policy may not be written to a public network", or "data under this policy may only be operated on in a trusted environment". When independently addressable data units have their own tags it becomes possible for larger data structures such as e.g. files to comprise a number of independently addressable data units having a number of different tags. This ensures the correct policy can be associated with a particular data unit irrespective of its location or association with other data in a memory structure, file structure or other data structure. The data management policy to be applied to data, and hence the tag, can be established in a number of ways.

(1) Data may already have a predetermined data management policy applied to it, and hence be associated with a pre-existing tag. When the NT kernel 250 makes a system call involving a piece of data, the data filter 240 checks for a pre-existing tag associated with that data, and if a pre-existing tag is present notifies the tag propagation module 220 to include the tag in the tag table 130, and to maintain the association of the tag with the data. Any tag associated with the data is maintained, and the data keeps its existing data management policy.

If there is no tag associated with the data, the following tag association methods can be used.

(2) Data read from a specific data source can have a predetermined data management policy corresponding to that data source applied to it. The data filter 240 checks for a data management policy corresponding to the specific data source, and if a predetermined policy does apply to data from that source notifies the tag propagation module 220 to include the corresponding tag in the tag table 130 and associate the tag with the data. For example, all data received over a private network from a trusted party can be associated with a tag indicative of the security status of the trusted party.

(3) When data has no pre-existing tag, and no predetermined data management policy applies to the data source from which the data originates, the tag management module 230 initiates an operating system function that allows a user to directly specify a desired data management policy for the data. The desired data management policy specified by the user determines the tag associated with the data. To ensure that the operating system function is authentic and not subject to subversion, it is desired that the operating system function of the tag management module 230 is trusted. This trust can be achieved and demonstrated to a user in a number of ways, as will be appreciated by the skilled person.

(4) Alternatively, when data has no pre-existing tag, and no predetermined data management policy applies to the

data source from which the data originates a default tag can be applied to the data.

Data management instructions are provided for subsequent instructions relating to internal processing of the tagged data. The data management instructions cause the tag propagation module 220 to maintain the association between the data and tag applied to it. Again, the data management instructions may include the instructions relating to internal processing of the data along with additional data management instructions. If the data is modified, e.g. by a logical or other operations, the relevant tag is associated with the modified data. Data management instructions for maintaining the association of tags with data as that data is manipulated and moved can be implemented using relatively simple state machine automatons. These automatons operate at the machine code level to effectively enforce the association and propagation of tags according to simple rules. For example, if data is moved the tag associated with the data at the move destination should be the same as the tag associated with the data before the move. In this simple example, any tag associated with the data at the move destination can be overwritten by the tag associated with the incoming data. Other automatons can be used to combine tags, swap tags, extend tags to other data, leave tags unchanged etc. dependent on the existing data tag(s) and type of operation to be carried out on the data.

The supervisor code 120 manages the tags in the tag table. A simple form of tag management comprises providing a data tag table that is large enough to accommodate a tag for each piece of tagged data. This results in a one-to-one

relationship between the data in the application memory space 110, and the data tags in the tag table, and a consequent doubling of the overall memory space required to run the application. However, memory is relatively cheap, and the one to one relationship enables simple functions to be used to associate the data with the relevant tag. As an alternative, different data structures can be envisaged for the data management information area, for example, a tag table can identify groups of data having a particular tag type. This may be advantageous when a file of data all associated with a single tag is involved in an operation. When more than one application is loaded in the user space 100, as shown in Figure 2 with the two application memory spaces 110A,110B, a shared tag table 130 can be used. As already mentioned, different tags can be applied to a separate data units within a file or other data structure. This allows an improved flexibility in subsequent manipulation of the data structure ensuring the appropriate policy is applied to the separate data units.

Data management instructions are also provided for instructions relating to writing of data outside the process. The data management instructions may include the instructions relating to writing of data outside the process along with other data management instructions. In this case, the data management instructions prompt the supervisor code 120 to notify the tag propagation module 220 of the tag associated with the data to be written. The system call to the NT kernel 250 is received by the data filter 240. The data filter 240 queries the allowability of the requested operation with the tag propagation module 220 to verify the tag associated with

the data to be written, and check that the data management policy identified by the tag allows the desired write to be performed with the data in question. If the desired write is within the security policy of the data in question, it is performed, with the data filter 240 controlling the file system driver 202 to ensure that the storage device drivers 203 to enforce the persistence of the tags with the stored data. If the data is not permitted to be written as requested, the write operation is blocked. Blocking may comprise writing random bits to the requested location, writing a string of zeros or ones to the requested location, leaving the requested location unaltered, or encrypting the data before writing.

A second example operating system data management architecture suitable for use in the computing platform of Figure 1 is shown in Figure 3. The example operating system data management architecture of Figure 3 relates to the Linux operating system.

Figure 3 shows a user space 100 and an OS kernel space 200. The user space 100 comprises application memory spaces 110A, 110B, supervisor code 120A,120B, and a tag table 130. The OS kernel space 200 comprises a tag propagation module 220, a tag management module 230, along with a Linux kernel 260 comprising an executable loader module 261, a process management module 262, a network support module 263 and a file system support module 264.

As the Linux operating system is open source, a number of the functions required to implement the data management system can be incorporated into the existing functional blocks of the kernel. In the example architectures of

Figure 3, the executable loader module 261, the process management module 262, the network support module 263 and the file system support module 264 are be modified versions of those included in a standard Linux kernel, as will be described below.

As before, the supervisor code 120 controls system calls, handles memory space tag propagation, and instructs policy checks in the OS kernel space 200 when required. Also as before, the tag propagation module 220 maintains policy information relating to allowable operations within the policies, and the tag management module 230 provides an administrative interface comprising an operating system function that allows a user to directly specify a desired data management policy for the data.

The operation of the Linux kernel 260 allows the data management architectures shown to carry out data flow control. The executable loader 261 includes a tagging driver that ensures applications are run under the control of the supervisor code 120. The process management module 262 carries out process management control to maintain the processor running the application or applications in a suitable state to enable tag association, monitoring and propagation. The network support module 263 enables the propagation of tags with data across a network, and the file system support module 264 enables the propagation of tags with data on disk. The network support module 263 and the file system support module 264 together provide the functionality of the data filter of Figure 2. Again, state machine based automation can be used to perform basic tag association, monitoring and propagation functions at a machine code level.

The modifications to the executable loader module 261, the process management module 262, the network support module 263 and the file system support module 264 can be easily implemented with suitable hooks.

Figure 4 shows a flow diagram outlining basic steps in an example method of operating system data management.

The method comprises a first step 300 of associating data management information with data input to a process; and a second step 310 of regulating operations involving the data input to the process in the first step 300 according to the data management information associated with the data in the first step 300. The basic first and second steps 300,310 are further expanded upon in the flow diagram of Figure 5.

Figure 5 shows a flow diagram outlining further steps in an example method of operating system data management.

The method of Figure 5 starts with an "external operation?" decision 312. If data on which the method is performed is read into memory space associated with a process from a location external to the memory space associated with the process, the outcome of the "external operation?" decision 312 is YES. Furthermore, if the data within the process is to be written to an external location, the outcome of the "external operation?" decision 312 is also YES. Following a positive decision at the "external operation?" decision, the method moves to the "tag present?" decision 314. Operations involving

data within the process result in a negative outcome at the "external operation?" decision 312.

At the "tag present?" decision 314, it is determined whether the data involved in the operation has data management information associated with it. If the data has no data management information associated with it, the association step 300 is performed, and the method returns to the "external operation?" decision 312.

In the association step 300, data management information is associated with the data in question. This association can be carried out by any of the methods described earlier, or by other suitable methods.

Following a positive decision at the "tag present?" decision 314, the method moves to the "operation allowed?" decision 316. At this decision, the data management information associated with the data is examined, and its compatibility with the specified external operation identified in the "external operation?" decision 312 is established.

If the data management information is compatible with the external operation, it is carried out in the execution step 318. Following the execution step 318, the method returns to the "external operation?" decision 312. Alternatively, if the data management information is not compatible with the external operation, it is blocked in the blocking step 318. Blocking in step 318 can comprise any of the methods described earlier, or by other suitable methods.

Any operations identified at the "external operation?" decision 312 as internal operations are carried out, with association of the data involved in the operation with the relevant data management information maintained in the tag propagation step 313.

Including the data management functionality with an operating system provides a first level of security, as operating system operation should be relatively free from security threatening bugs compared to either commercial or open source application software. Furthermore, if the operating system allows trusted operation after a secure boots, for example as provided for by the Trusted Computing Platform Alliance (TCPA) standard, the data management functionality can also form part of the trusted system. This enables the data management functions to also form part of the trusted system, enabling e.g. digital rights management or other secrecy conditions to be enforced on data.

It is possible that the computing platform for operating system data management could refuse to open or write data with a pre-existing tag unless the computing platform is running in a trusted mode, adding to the enforceability of data flow control under the data management system. This is particularly useful when encrypted data is moved between trusted computing platforms over a public network.

An operating system data management method, and a computing platform for operating system data management have been described. The data management method and computing platform allow a supervisor code to monitor data flow into and out of an application using data management

information. As data is used within an application process, the data management information is propagated with the data. This allows the supervisor code to ensure that only external write operations which are compatible with a data management policy for the data are performed. The data flow monitoring and enforcement enabled by the data management method and computing platform facilitate the construction of systems that support digital rights management and other data privacy functions, but avoid the problems associated with system wide approaches to data flow control systems. In particular, the granularity provided by associating data management information with data units that are individually addressable rather than with a data structure such as a file of which the individually addressable data units are part offers improved flexibility in how security is enforced. The method and computing platform described do not require source code modification of application and subsequent recompilation. Furthermore, the method and system described can easily be retrospectively implemented in a variety of known operating systems, for example Windows NT and Linux as show herein.

The functionality described above can also be implemented on a virtual machine.

There will now be described a method and apparatus for handling tagged data. These are applicable to the data tagged and propagated as described above as well as to data tagged in other ways, for instance at the file level (i.e. all data in a file having the same tag).

Figure 6 of shows a data handling apparatus 400 forming a part of the computing platform 1 shown in Figure 1. The data handling apparatus 400 comprises a system call monitor 402, a tag determiner 404 and a policy interpreter 406. The policy interpreter 406 comprises a policy database 408 and a policy reconciler 410. Also shown in Figure 6 are external devices indicated generally at 412, which can be local external devices 414 such as printers, CD writers, floppy disk drives, etc or any device on a network (which can be a local network, a wide area network or a connection to the Internet), such as a printer, another computer, CD writer, etc. The data handling apparatus 400 can be embodied in hardware or software, and in the latter case may be a separate application or more preferably runs at an operating system level.

Operation of the apparatus shown in Figure 6 is explained with reference to Figure 7 which shows a functional flow diagram thereof.

In step 450 the data handling apparatus 400 runs on a computing platform 1 and the system call monitor 402 checks each system call at the kernel layer of the operating system to determine whether it is a system call in relation to which the data handling apparatus 400 is configured to control. Typically the controlled system calls are those involving writes of data to devices (which include writes to network sockets) so that the transfer of data externally of the operating system and computing platform memory can be controlled. The system call monitor 402 implemented at the kernel level keeps track of new file descriptors being created during the process execution that refer to controlled external devices and

network sockets. The system call monitor 402 also monitors all system calls where data is written to these file descriptors. Whenever a system call is intercepted that causes data write or send, the process is stopped and both the data and the file descriptor that this data is being written/sent to are examined. The system call monitor 402 has a list of predetermined system calls that should always be denied or permitted. If the intercepted system call falls into this category the system call monitor uses this fast method to permit or deny a system call. If the fast method cannot be used, the system call monitor needs to ask the policy interpreter 406 in user space for a policy decision. Thus either the system call monitor 402 or the tag determiner 404 and policy interpreter 406 can be a means for applying a data handling policy to the system call upon a predetermined system call being detected

Once a predetermined system call has been detected by system call monitor 402, then in step 452 the tag determiner 404 determines what security tag or tags are associated with the corresponding operation. For the purpose of this explanation of an embodiment of the present invention, it is assumed the system call is of data from a file to a networked device. Using the data tagging described above, a plurality of tags will apply. Using other tagging techniques there may only be one tag associated with a file. For this embodiment it is assumed that there are several tags associated with the data. The tags associated with the data relevant to the action of the system call are communicated to the policy interpreter 406 in step 454.

In step 456, the policy interpreter 406 determines the policy to be applied to the data. Referring to Figure 8, the sub-steps of step 456 are shown in more detail. In step 458 a policy for each tag is looked up from the policy database 408. Since the so determined policies may be inconsistent, the resultant policies are supplied to policy reconciler 410, which in step 460 carries out a policy reconciliation to generate a policy to apply to the data. The nature of the policy reconciliation is a matter of design choice for a person skilled in the art. At its simplest policy reconciliation will provide that the most restrictive policy derived from all restrictions and requirements of the policies associated with the tags applies, effectively ANDing all the policies. However, many alternatives exist. The policy reconciler may make policy determinations based on the intended destination of the relevant data, which is known from information provided by the system call monitor 402.

Once a reconciled policy has been determined by policy reconciler 410, this is the output from policy interpreter 406 that is returned to system call monitor 402. The system call monitor allows the stopped process to continue execution after it applies the result to the operation in question in step 462 (Figure 7).

Generally there will be three policy applications. The first will be to permit the operation. The second will be to block the operation. The third will be to permit the operation but to vary it in some way. The main variation is the encryption of the data being transmitted for additional security.

In any data transmission, tags may be propagated as described above.

The reader's attention is directed to all papers and documents which are filed concurrently with or previous to this specification in connection with this application and which are open to public inspection with this specification, and the contents of all such papers and documents are incorporated herein by reference.

All of the features disclosed in this specification (including any accompanying claims, abstract and drawings), and/or all of the steps of any method or process so disclosed, may be combined in any combination, except combinations where at least some of such features and/or steps are mutually exclusive.

Each feature disclosed in this specification (including any accompanying claims, abstract and drawings), may be replaced by alternative features serving the same, equivalent or similar purpose, unless expressly stated otherwise. Thus, unless expressly stated otherwise, each feature disclosed is one example only of a generic series of equivalent or similar features.

The invention is not restricted to the details of the foregoing embodiment(s). The invention extends to any novel one, or any novel combination, of the features disclosed in this specification (including any accompanying claims, abstract and drawings), or to any novel one, or any novel combination, of the steps of any method or process so disclosed.

Claims

1. A data handling apparatus for a computer platform using an operating system, the apparatus comprising a system call monitor for detecting predetermined system calls, and means for applying a data handling policy to the system call upon a predetermined system call being detected.

2. A data handling apparatus according to claim 1, in which the policy is to require the encryption of at least some of the data.

3. A data handling apparatus according to claim 1 or claim 2, in which a policy interpreter in its application of the policy automatically encrypts the at least some of the data.

4. A data handling apparatus according to any preceding claim, in which predetermined system calls are those involving the transmission of data externally of the computing platform.

5. A data handling apparatus according to any preceding claim, in which the means for applying a data handling policy comprises a tag determiner for determining any security tags associated with data handled by the system call, and a policy interpreter for determining a policy according to any such tags and for applying the policy.

6. A data handling apparatus according to claim 5, in which the policy interpreter is configured to use the

intended destination of the data as a factor in determining the policy for the data.

7. A data handling apparatus according to claim5 or claim 6, in which the policy interpreter comprises a policy database including tag policies and a policy reconciler for generating a composite policy from the tag policies relevant to the data.

8. A data handling apparatus according to any preceding claim, in which the computing platform comprises a data management unit, the data management unit arranged to associate data management information with data input to a process, and regulate operating system operations involving the data according to the data management information.

9. A data handling apparatus according to claim 8, in which the computing platform further comprises a memory space, and is arranged to load the process into the memory space and run the process under the control of the data management unit.

10. A data handling apparatus according to claim 8 or claim 9, in which the data management information is associated with at least one data sub-unit as data is input to a process from a data unit comprising a plurality of sub-units.

11. A data handling apparatus according to any one of claims 8-10, in which data management information is associated with each independently addressable data unit.

12. A data handling apparatus according to any one of claims 8-11, in which the data management unit comprises part of an operating system kernel space.

13. A data handling apparatus according to claim 12, in which the operating system kernel space comprises a tagging driver arranged to control loading of a supervisor code into the memory space with the process.

14. A data handling apparatus according to claim 13, in which the supervisor code controls the process at run time to administer the operating system data management unit.

15. A data handling apparatus according to claim 14, in which the supervisor code is arranged to analyse instructions of the process to identify operations involving the data, and, provide instructions relating to the data management information with the operations involving the data.

16. A data handling apparatus according to any one of claims 13-15, in which the memory space further comprises a data management information area under control of the supervisor code arranged to store the data management information.

17. A data handling apparatus according to any one of claims 8-16, in which the data management unit comprises a data filter to identify data management information associated with data that is to be read into the memory space.

18. A data handling apparatus according to any one of claims 8-17, in which the data management unit further comprises a tag management module arranged to allow a user to specify data management information to be associated with data.

19. A data handling apparatus according to any one of claims 8-18, in which the data management unit comprises a tag propagation module arranged to maintain an association with the data that has been read into the process and the data management information associated therewith.

20. A data handling apparatus according to claim 19, in which the tag propagation module is arranged to maintain an association between an output of operations carried out within the process and the data management information associated with the data involved in the operations.

21. A data handling apparatus according to claim 19 or claim 20, in which the tag propagation module comprises state machine automatons arranged to maintain an association between an output of operations carried out within the process and the data management information associated with the data involved in the operations.

22. A data handling method for a computer platform using an operating system, the method comprising the steps of: detecting predetermined system calls, and applying a data handling policy to the system call upon a predetermined system call being detected.

23. A data handling method according to claim 22, in which the policy is to require the encryption of at least some of the data.

24. A data handling method according to claim 23, in which in its application of the policy at least some of the data is automatically encrypted.

25. A data handling method according to any one of claims 22-24, in which predetermined system calls are those involving the transmission of data externally of the computing platform.

26. A data handling method according to any one of claims 22-25, in which the method includes the steps of: determining any security tags associated with data handled by the system call, determining a policy according to any such tags and applying the policy.

27. A data handling method according to claim 26, in which a composite policy is generated from the tag policies relevant to the data.

28. A data handling method according to claim 26 or claim 27, in which the intended destination of the data is used as a factor in determining the policy for the data.

29. A data handling method according to any one of claims 22-28 in which the method further comprises the steps of: (a) associating data management information with data input to a process; and (b) regulating operating system operations involving the data according to the data management information.

30. A data handling method according to claim 29, in which supervisor code administers the method by controlling the process at run time.

31. A data handling method according to claim 29 or claim 30, in which the step (a) comprises associating data management information with data as the data is read into a memory space.

32. A data handling method according to any one of claims 29-31, in which the step (a) comprises associating data management information with at least one data sub-unit as data is read into a memory space from a data unit comprising a plurality of data sub-units.

33. A data handling method according to any one of claims 29-32, in which the step (a) comprises associating data management information with each independently addressable data unit that is read into the memory space.

34. A data handling method according to any one of claims 29-33, in which the data management information is written to a data management memory space under control of the supervisor code.

35. A data handling method according to claim 34, in which the supervisor code comprises state machine automatons arranged to control the writing of data management information to the data management memory space.

36. A data handling method according to any one of claims 29-35, in which the step (b) comprises sub-steps (b1)

identifying an operation involving the data; (b2) if the operation involves the data and is carried out within the process, maintaining an association between an output of the operation and the data management information; and (b3) if the operation involving the data includes a write operation to a location external to the process, selectively performing the operation dependent on the data management information.

37. A data handling method according to claim 36, in which, the step (b1) comprises: analysing process instructions to identify operations involving the data; and, providing instructions relating to the data management information with the operations involving the data.

38. A data handling method according to any one of claims 29-37, in which the process instructions are analysed as blocks, each block defined by operations up to a terminating condition.

39. A computer program for controlling a computing platform to operate in accordance with any one of claims 22-38.

40. A computer platform configured to operate according to any one of claims 22-38.

41. A data handling apparatus substantially as described herein, with reference to the drawings that follow.

42. A data handling method substantially as described herein, with reference to the drawings that follow.

Abstract

Improvements in and Relating to Data Handling Apparatus and Methods

A data handling apparatus (400) for a computer platform (1) using an operating system, the apparatus comprising a system call monitor (402) for detecting predetermined system calls, and means (402, 404, 406) for applying a data handling policy to the system call upon a predetermined system call being detected. A corresponding method is disclosed.

Figure 6

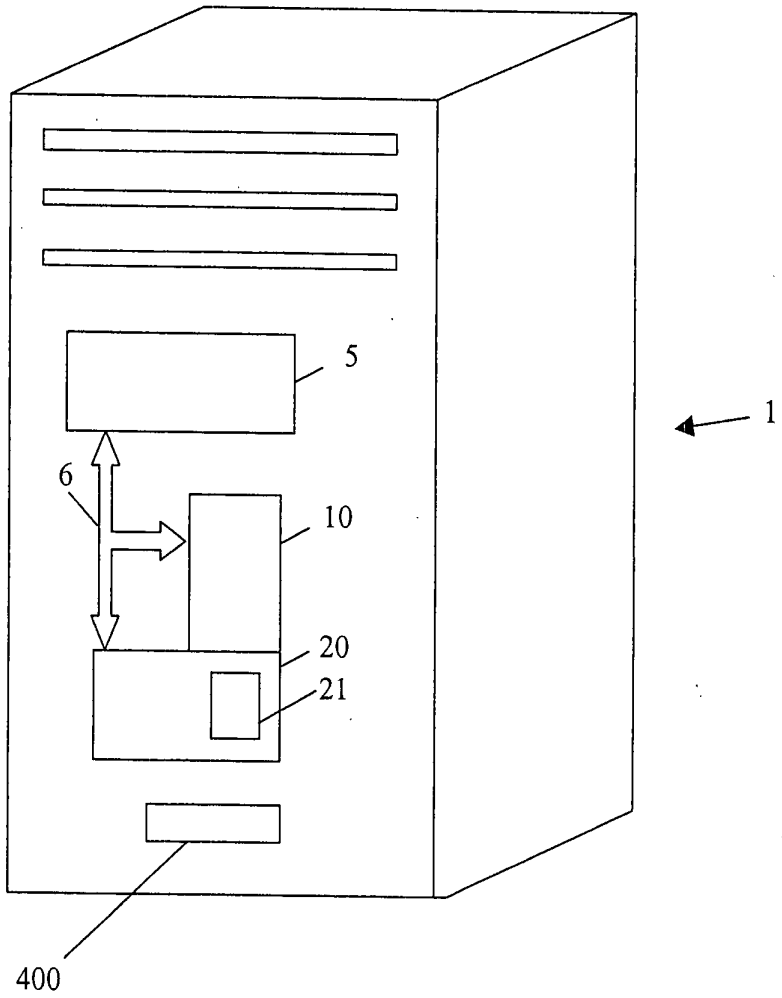


Figure 1

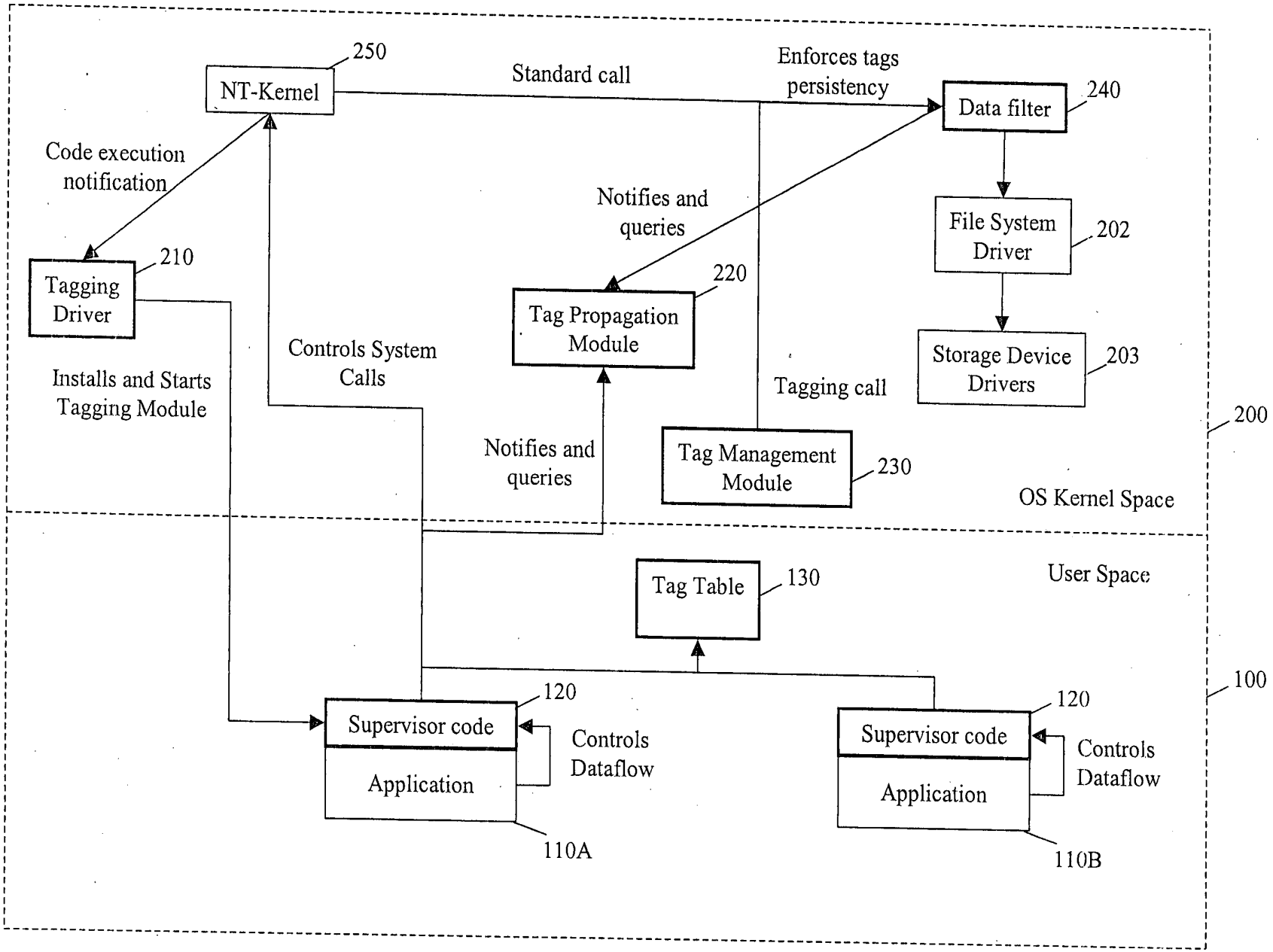

NT-Kernel
250
Standard call
Enforces tags persistency
Data filter
240
Code execution notification
Notifies and queries
File System Driver
202
Tagging Driver
210
220
Tag Propagation Module
Storage Device Drivers
203
Controls System Calls
Installs and Starts Tagging Module
Tagging call
Notifies and queries
Tag Management Module
230
200
OS Kernel Space
Tag Table
130
User Space
100
120
120
Supervisor code
Controls Dataflow
Supervisor code
Controls Dataflow
Application
Application
110A
110B

Figure 2

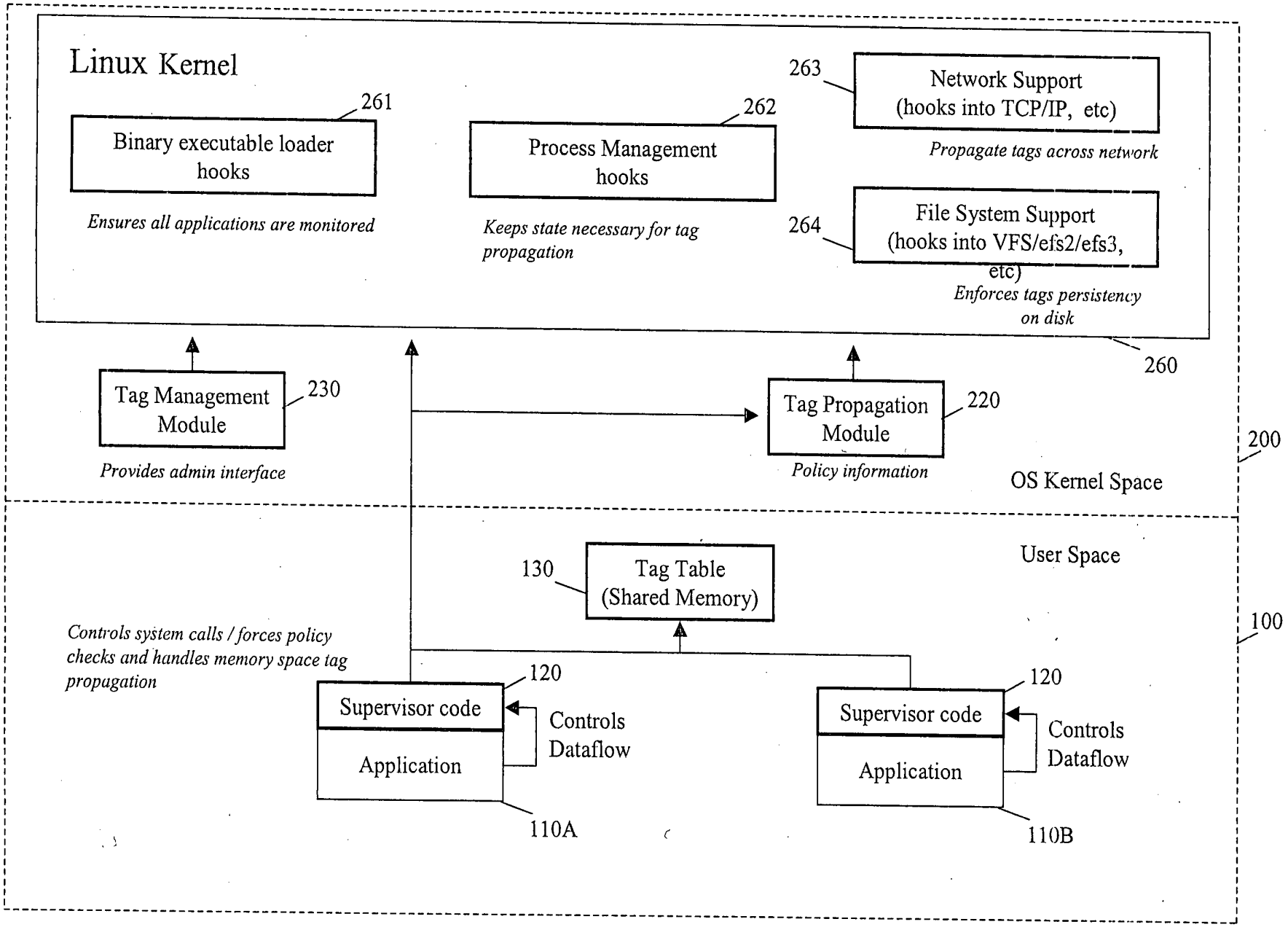

Linux Kernel
261
Binary executable loader hooks
Ensures all applications are monitored
262
Process Management hooks
Keeps state necessary for tag propagation
263
Network Support (hooks into TCP/IP, etc)
Propagate tags across network
264
File System Support (hooks into VFS/efs2/efs3, etc)
Enforces tags persistency on disk
260
Tag Management Module
230
Provides admin interface
Tag Propagation Module
220
Policy information
200
OS Kernel Space
User Space
130
Tag Table (Shared Memory)
Controls system calls / forces policy checks and handles memory space tag propagation
120
Supervisor code
Application
Controls Dataflow
110A
120
Supervisor code
Application
Controls Dataflow
110B
100

Figure 3

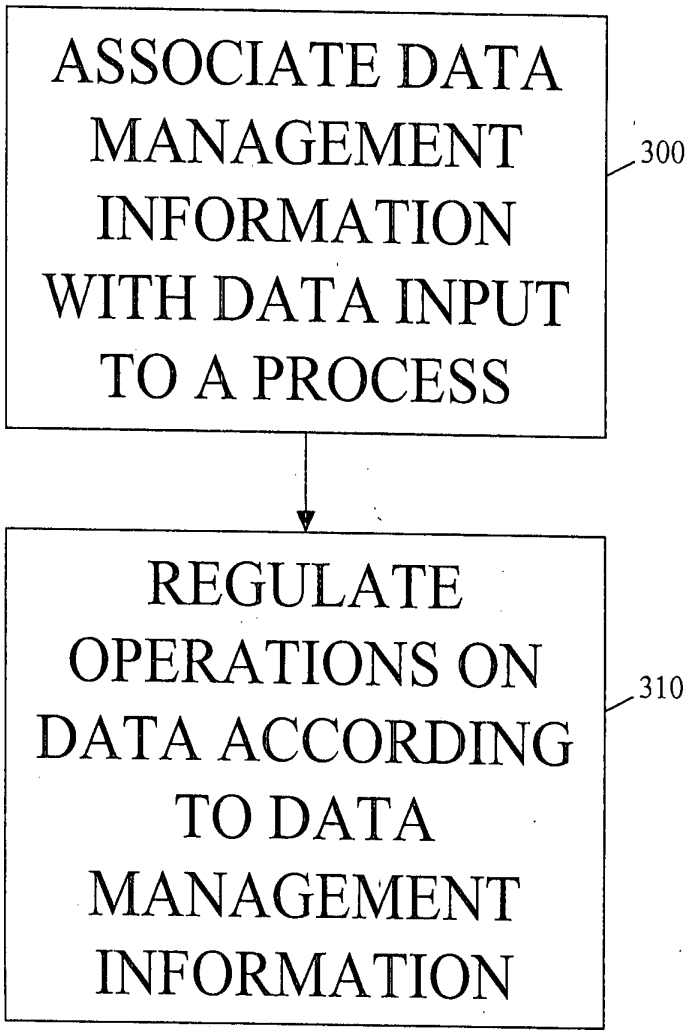

ASSOCIATE DATA MANAGEMENT INFORMATION WITH DATA INPUT TO A PROCESS
300
REGULATE OPERATIONS ON DATA ACCORDING TO DATA MANAGEMENT INFORMATION
310

Figure 4

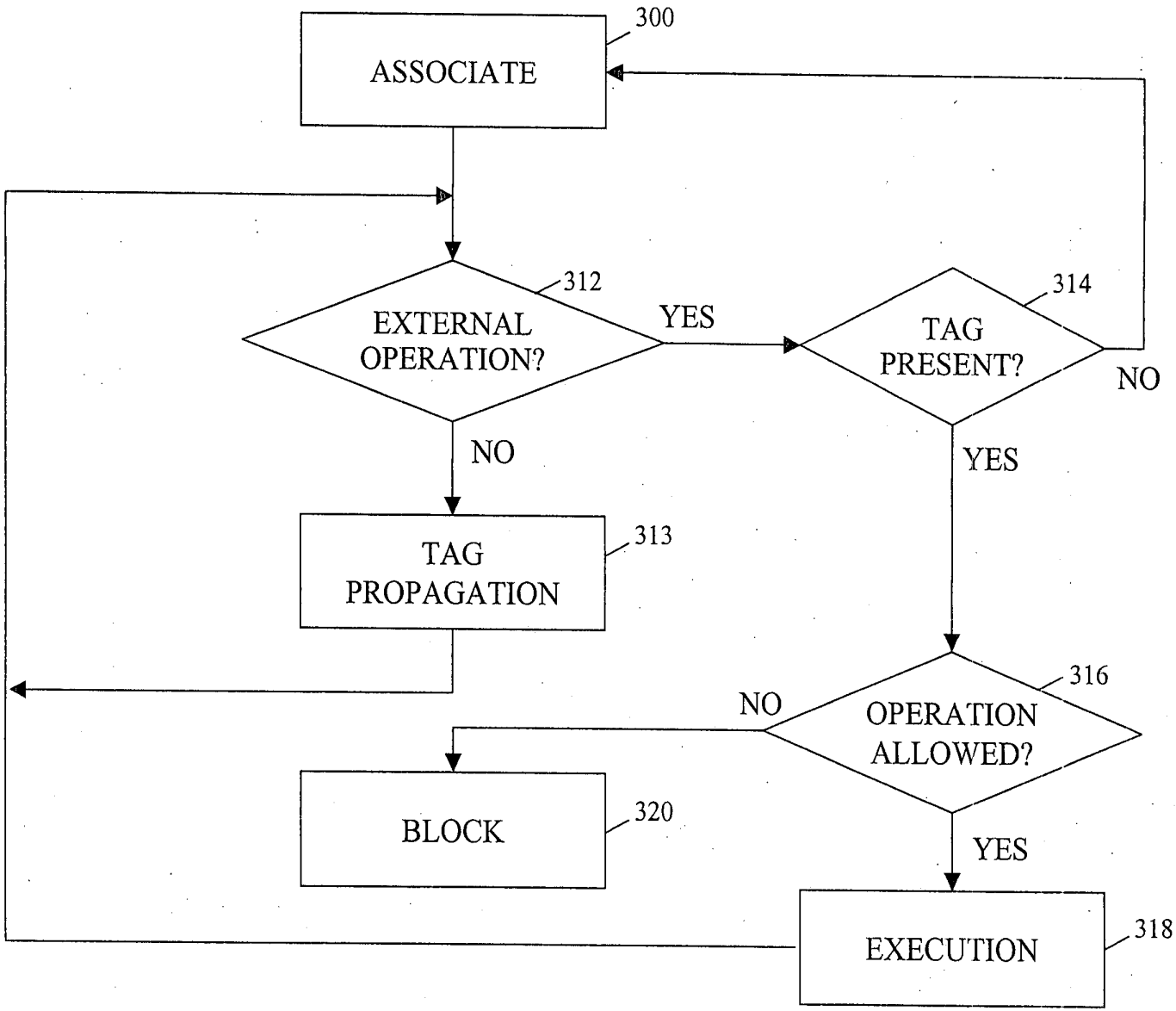

ASSOCIATE
300
EXTERNAL OPERATION?
312
YES
TAG PRESENT?
314
NO
NO
YES
TAG PROPAGATION
313
NO
OPERATION ALLOWED?
316
BLOCK
320
YES
EXECUTION
318

Figure 5

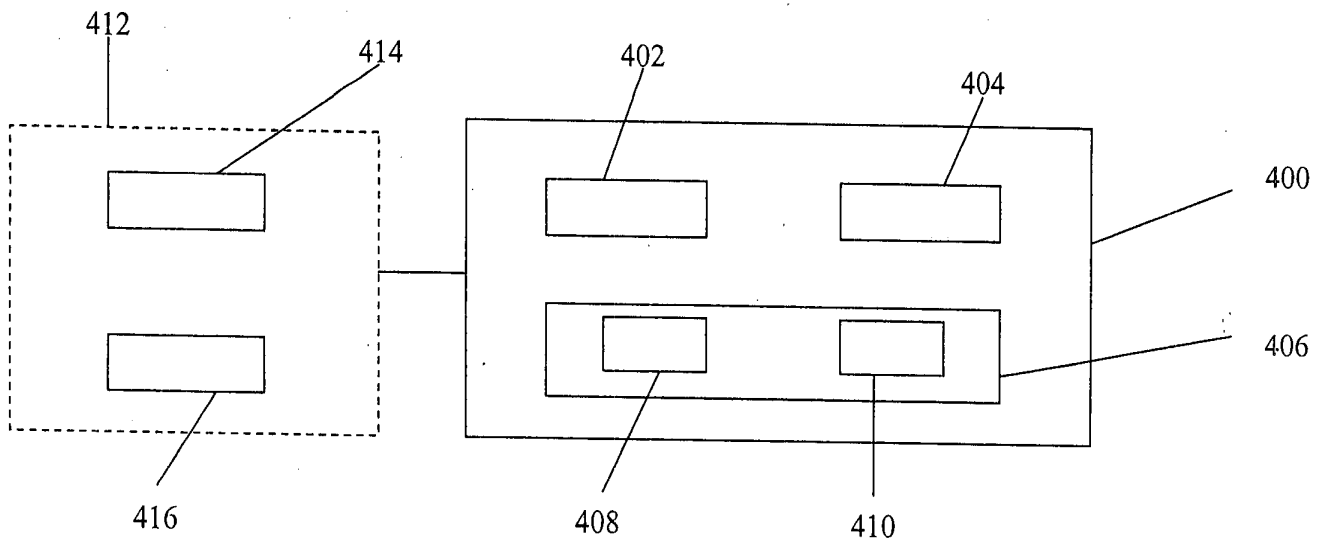


Figure 6

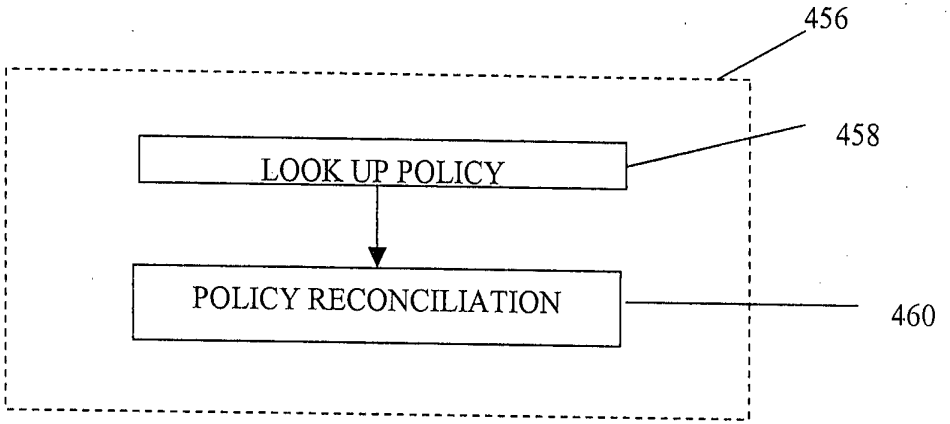


Figure 8

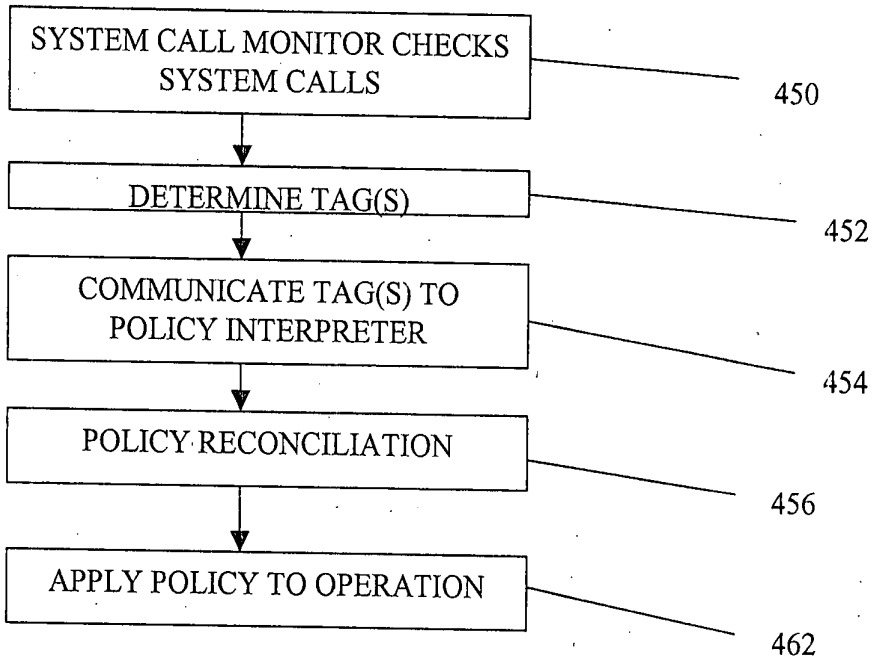

SYSTEM CALL MONITOR CHECKS SYSTEM CALLS
450
DETERMINE TAG(S)
452
COMMUNICATE TAG(S) TO POLICY INTERPRETER
454
POLICY RECONCILIATION
456
APPLY POLICY TO OPERATION
462

Figure 7